Technische Universität Chemnitz
Fakultät für Mathematik
Professur Mathematik in Industrie und Technik

Sommersemester 2009
Prof. Dr. Peter Benner
Dipl.-Math. Jens Saak
Dipl.-Ing. Martin Bernauer

Numerische Mathematik– 13. Hausaufgabe

Abgabetermin: 8./9.7.2009
(in der jeweiligen Übungsgruppe)

Theoretische Aufgaben

Aufgabe 1 (4 Punkte) (Diskrete Cosinus-Transformation)

Beim JPEG-Verfahren zur Bildkomprimierung werden die Bilddaten *vor* der Komprimierung entsprechend aufbereitet, damit eine stärkere Kompression der modifizierten Daten möglich wird. Dabei werden Farbwerte in einige wenige diskrete Frequenzwerte umgewandelt – ein Schritt, der im Falle von Rundungen für den Qualitätsverlust bei der Komprimierung verantwortlich ist. Hochfrequente Anteile entsprechen kleinen Farbnuancen, die das menschliche Auge kaum unterscheiden kann.
Die Grundlage für dieses Verfahren bietet die Diskrete Cosinus-Transformation (DCT):

$$F_k = \sum_{j=0}^{N-1} f_j \cos\left(\frac{\pi k(j+\frac{1}{2})}{N}\right), \qquad k = 0, \ldots, N-1 \tag{1}$$

(f_k sind die Originaldaten, F_k die entsprechenden Frequenzanteile). Die Rücktransformation (iDCT) dazu lautet

$$f_j = \frac{2}{N}\left(\frac{F_0}{2} + \sum_{k=1}^{N-1} F_k \cos\left(\frac{\pi k(j+\frac{1}{2})}{N}\right)\right), \qquad j = 0, \ldots, N-1. \tag{2}$$

Die Anwendung dieser Transformation ist notwendig, weil Bilddaten weder periodisch sind noch komplexe Anteile enthalten.

a) Durch „Spiegelung“ von f zu $\tilde{f} = (f_0, \ldots, f_{N-1}, f_{N-1}, \ldots, f_0)^\top$, d.h.

$$\tilde{f}_j = \begin{cases} f_j, & j = 0, \ldots, N-1 \\ f_{2N-j-1}, & j = N, \ldots, 2N-1 \end{cases}$$

kann man eine modifizierte diskrete Fourier-Transformation doppelter Länge folgendermaßen erhalten:

$$\tilde{F}_k = \sum_{j=0}^{2N-1} \tilde{f}_j \exp\left(\frac{2\pi(j+\frac{1}{2})k}{2N}\, i\right), \tag{3}$$

bzw.

$$\tilde{f}_j = \frac{1}{2N} \sum_{k=0}^{2N-1} \tilde{F}_k \exp\left(\frac{-2\pi(j+\frac{1}{2})k}{2N}\, i\right). \tag{4}$$

Zeigen Sie, dass die ersten N Komponenten $\tilde{F}_k$ ($k = 0, \ldots, N-1$) bis auf einen konstanten Faktor mit den F_k aus der Cosinus-Transformation (1) übereinstimmen. Wie lautet dieser konstante Faktor?
Welche Werte erhält man für $\tilde{F}_N$ und $\tilde{F}_{N+1}, \ldots, \tilde{F}_{2N-1}$?

b) Um nun die schnelle Fourier-Transformation (FFT) anwenden zu können, müssen Sie noch geeignete Faktoren γ_k finden, so dass für $\tilde{g}_k = \gamma_k \tilde{F}_k$ aus den modifizierten Transformationen aus (3) und (4) die „normalen“ Fourier-Transformationen entstehen:

$$\tilde{g}_k = \sum_{j=0}^{2N-1} \tilde{f}_j \exp\left(\frac{2\pi jk}{2N}\, i\right), \qquad \tilde{f}_j = \frac{1}{2N} \sum_{k=0}^{2N-1} \tilde{g}_k \exp\left(\frac{-2\pi jk}{2N}\, i\right).$$

Bestimmen Sie diese γ_k.

Bemerkung: Wir besprechen die konkrete Anwendung in der Bildverarbeitung (2D-DCT) in der Übungsstunde am 15./16. 7.

Aufgabe 2 (4 Punkte) (Berechnung von Splines)

Leiten Sie für die Wertepaare

x_i	-2	-1	0	1	2
y_i	0.2	0.5	1	0.5	0.2

die Systeme zur Bestimmung der Momente $M_0, \ldots, M_4$ für

a) natürliche,

b) periodische und

c) vollständige

Splines her. Verwenden Sie im letzten Fall die Bedingungen $f'(-2) = 1$ und $f'(2) = 0$.

Programmieraufgaben

Aufgabe 1 (3 Punkte) (Umsetzung der diskreten Cosinus-Transformation)

Implementieren Sie mit Hilfe von `fft` einen $O(n \cdot \log_2 n)$-Algorithmus `F=fct(f)` zur Berechnung der eindimensionalen DCT.
Hinweise:

- Die Matlab-Funktion `fft(X)` berechnet für eine $n \times m$-Matrix `X` in jeder Spalte eine FFT der Länge n. Versuchen Sie, Ihre Funktion `fct(f)` ebenso zu implementieren.
- Eventuelle imaginäre Anteile, die durch Anwendung von `fft` (wegen Rundungsfehlern) entstehen können, sollten sich lediglich als $\pm$`0.0000i` zeigen und dürfen in diesem Fall am Ende mit `real(F)` „entfernt" werden.
- Wenn Sie die Theorie-Aufgabe 1 nicht gelöst haben oder Ihrem Ergebnis misstrauen, implementieren Sie die Funktion `dct(f)` nach Gleichung (1). Bei `fct(ones(n,1))` z.B. sollte der Vektor `[n,0,...,0]'` herauskommen.

Zusatzaufgabe (2 Punkte)
Implementieren Sie auch die inverse Cosinus-Transformation, `f=ifct(F)`, nach (2) bzw. (4). Überprüfen Sie, ob nach Ihrer Implementation `X=ifct(fct(X))` gilt.

Aufgabe 2 (3 Punkte) (Berechnung von Splines)

Schreiben Sie ein MATLAB-Programm zur Berechnung

a) natürlicher,

b) periodischer und

c) vollständiger (vorgegebene Ableitung in a und b)

kubischer Splines. Das Programm soll als Eingabeparameter den Stützstellenvektor x mit den zugehörigen Werten f haben (als optionaler Eingabeparameter für den Fall c) sehen Sie einen Vektor mit den Ableitungen in a und b vor), und die Parameter a_j, b_j, c_j, d_j, $j = 1, \ldots, n$, als Vektoren zurückgeben.
Testen Sie ihr Programm mit den Daten

	x_0	x_1	x_2
	0	1	2
a)	0	1	8
b)	8	1	8
c)	2	1	8

wobei im letzten Fall $f'(0) = 0$ und $f'(2) = 3$ vorgegeben sind. Vergleichen Sie im Fall a) alle drei Varianten mit den Daten $f'(0) = f'(2) = 0$ im Falle periodischer Splines. Plotten Sie für alle Testfälle die entsprechenden Ergebnisse auf dem Gitter $[0 : 0.1 : 2]$.